BBC RD 1996/11

Research and Development Report

MOTION ESTIMATION FOR MPEG-2 CODING USING 'TRUE' VECTORS

G.A. Thomas, M.A., Ph.D., C.Eng., M.I.E.E.

Research and Development
Policy and Planning
THE BRITISH BROADCASTING CORPORATION

BBC RD 1996/11

MOTION ESTIMATION FOR MPEG-2 CODING USING 'TRUE' VECTORS

G.A. Thomas, M.A., Ph.D., C.Eng., M.I.E.E.

Summary

The motion estimator is a key component of any MPEG-2 video coder. It is common practice to use a method based on block-matching, which generates vectors designed to yield good predictions of images to be coded, without regard for whether the vectors correspond to actual motion. This Report describes a method of deriving motion vectors for an MPEG-2 coder from a motion estimator designed to produce 'true' motion vectors for other broadcast applications, such as standards conversion. The performance of these vectors is compared to those from full-search block-matching. It is found that the coding efficiency of both types of vectors is similar for most kinds of picture material, but the 'true' vectors give superior performance for picture material such as fades, where differences between successive images can be misinterpreted as motion. Other advantages include the ability to re-use vectors generated by up-stream processing, offering a potential saving in hardware. Furthermore, the approach described in this Report allows rapid movement to be measured and accounted for in a hardware-efficient manner.

This work was carried out as a part of the ***COUGAR*** *project, a part of the EU-funded* ***RACE*** *programme.*

Issued under the Authority of

Research & Development
Policy & Planning
BRITISH BROADCASTING CORPORATION

Head of Research & Development

(R034)

1996

MOTION ESTIMATION FOR MPEG-2 CODING USING ‘TRUE’ VECTORS

G.A. Thomas, M.A., Ph.D., C.Eng., M.I.E.E.

MOTION ESTIMATION FOR MPEG-2 CODING USING ‘TRUE VECTORS

G.A. Thomas, M.A., Ph.D., C.Eng., M.I.E.E.

1. INTRODUCTION

Most previous work concerned with the development of MPEG-2 coding techniques has used the criterion of minimising the prediction error when selecting motion vectors and prediction modes. Such an approach helps to minimise the total coding error (often measured as the signal-to-noise ratio). However, this takes no account of the subjective appearance of the coding artefacts that are introduced, which can vary significantly between different modes and vectors. The ‘motion’ vectors produced may be of little use in related applications, such as up-conversion for display, or error concealment in the decoder.

This Report considers the use of high-quality motion estimation methods to provide vectors for an MPEG-2 coder. The work described used a software model of a commercially-available motion estimator that has been proven in the field of standards conversion. It is based on the phase correlation method[1,2], and is known to produce vectors that are a good approximation to true motion.

This motion estimator, in common with many others designed for applications outside MPEG-2, estimates motion between successive fields, rather than between the pairs of fields or frames within the group-of-pictures (GOP) required for the various prediction modes. Furthermore, the vectors are generated for individual pixels, rather than blocks of 16 × 16 pixels (macroblocks). Possible ways of converting the vector information into the required form are described, and the results of computer simulations are presented.

In addition to obviating the need to design a new motion estimator specifically for an MPEG-2 coder, the vector conversion method that has been developed offers other benefits. Since almost *any* inter-field motion estimator of a reasonable quality may be used as a source of vectors, the possibility exists for using a vector signal derived in up-stream processing, such as noise reduction or standards conversion. By using a common vector signal in cascaded processing and coding operations, the possibility of degradations being introduced by the selection of different vectors in successive operations is reduced. With some sources of video signal, a source of ‘perfect’ vectors may be available as a by-product of the video generation (for example, information from camera motion sensors in a virtual studio); such a vector signal could be used directly to control the MPEG-2 coder. It is also possible to extract the vectors from an MPEG-2 bitstream for use in operations carried out on the signal after decoding. This may be of particular benefit if the vectors used for coding represent a close approximation to the true motion. A method for recovering vectors from a coded bitstream and their use for display up-conversion, are presented in another Report[3].

The structure of this Report is as follows. The algorithm that has been developed for converting inter-field to GOP-based vectors is described in Section 2, and simulation results are presented. Modifications to the vector refinement stage of the algorithm, to improve its performance on fades and dissolves, are presented in Section 3. Detailed consideration is given to half-pixel refinement in Section 4. The main conclusions of the work are summarised in Section 5.

2. VECTOR TRACING

2.1 Description of the algorithm

The most difficult aspect of the conversion process from pixel-based inter-field vectors to macroblock-based ‘GOP’ vectors is the need to change the time period between the fields related by the vectors. The MPEG-2 standard requires vectors to predict P (predicted) pictures from preceding P or I (intra-coded) pictures; the period separating these is typically three to four frames. Vectors are also required to predict B (bi-directional) pictures from preceding and following P or I pictures. In either case, two types of vector are required: ‘field’ and ‘frame’. The field vectors relate blocks (in the field that is being predicted) to the matching regions in the field from which the prediction is to be made. These are respectively referred to as the ‘predicted’ and ‘reference’ fields. The frame vectors refer to both fields together, and are used in ‘frame’ prediction mode, where both fields are thought of as comprising a single progressively-scanned image.

A method for carrying out this conversion is described in the following paragraphs. The method requires two motion vector signals; one, relating samples in the present field to samples in the preceding field (‘backward’ vectors), and the other, relating samples in the present field to samples in the following field (‘forward’ vectors). It is assumed that one vector is provided for each pixel in the image. This is the form of the vectors provided by the commercially-available phase correlation motion estimation hardware[2].

The conversion method involves ‘tracing’ vectors from field to field, and was first described in Ref. 4. A similar approach has previously been suggested as a means for initialising a full-search block-matching algorithm[5]. The forward-looking vector field is used when tracing forwards in time; similarly, the backward-looking field is used when tracing backwards. For example, the way in which ‘same-field-polarity forward vectors’ (in MPEG-2 terminology) would be derived for a P picture is illustrated in Fig. 1. This can be thought of as a concatenation of five tracing processes, each of which extends the vector back by one more field period, to give a vector relating a field of the P picture, to the reference field of the I picture six fields earlier. The converse of this would be the case if prediction from a *future* field was required; each process would work forwards in time, using forward-pointing vectors. Consider the first field of the P picture in Fig. 1. For each pixel in this field, the first tracing process computes the sum of the vector assigned to this pixel and the vector assigned to the pixel in the preceding field nearest to where it points. The shortest dashed arrow shows the resulting vector sum. The process is repeated four more times, each starting with the traced vector from the previous operation, and producing vectors shown by other dashed arrows. The final result, shown as a thick dashed arrow, is the required vector (which indicates which region of the preceding I picture to use to predict the P picture). The entire process is repeated, starting at the second field of the P picture, to derive the vectors for predicting this field (for clarity, the intermediate results are not shown for this). The tracing method can be compared to the conventional method of motion estimation for MPEG-2, in which vectors are chosen by direct comparison between predicted and reference fields.

The above process can be applied to every pixel in the image to obtain a pixel-rate vector field pointing forwards or backwards over any desired time period. From this, a vector for every macroblock can be derived; for example, by averaging all vectors within the macroblock, or by selecting the most frequently-occurring vector. A more computationally-efficient alternative is to trace a single vector per macroblock, by moving the pixel-to-macroblock conversion process to **before** the tracing stage. This significantly reduces the number of vectors that need to be traced. The two approaches might sometimes be expected to give different answers at object boundaries; for example, if the vector for the pixel in the centre of the block did not correspond to the dominant motion in the block.

Whilst vector tracing provides a convenient method for converting from any pixel-based vector field to one appropriate for MPEG coding, it would be expected to show some problems. In particular, the accuracy of the traced motion vectors would be expected to be reduced from that of the original inter-field vectors, since any inaccuracies in the vector field will accumulate as the vectors are integrated. It was therefore envisaged that some form of vector ‘refinement’ may be needed to correct for inaccuracies of the order of a few pixels. This process could make direct comparisons between the predicted and reference fields. It could be implemented, for example, using a conventional block-matching method with a very small search range, using the *traced vector* as a starting point. Such a process is sometimes called a directed blockmatch.

Fig. 2 shows block diagrams of the processes involved for both the ‘full-resolution’ tracing approach, and the more computationally-efficient method where the ‘representative’ vector for each macroblock is selected before, rather than after, tracing. The refinement processes are also shown. Note that the diagrams only show the derivation of the vectors for forward prediction; a second set of processing is required to generate backward vectors for use in ‘B’ pictures. In both cases, the tracing process computes the four vectors needed for field-type prediction of both fields of the frame being coded (relating the odd and even fields in this frame to the odd and even fields in the reference frame). This requires two separate tracing operations for each frame; one starting from the odd field of the frame being coded, and the other starting from the even field. Each tracing process works backwards to the later of the two fields of the reference frame to yield one field vector, and then works backwards over one more field to produce vectors pointing to the earlier field. The tracing algorithm cannot directly produce vectors for frame-type prediction, since the incoming vectors relate fields rather than frames. The vector for frame prediction is instead derived simply by averaging the two traced vectors that relate fields of

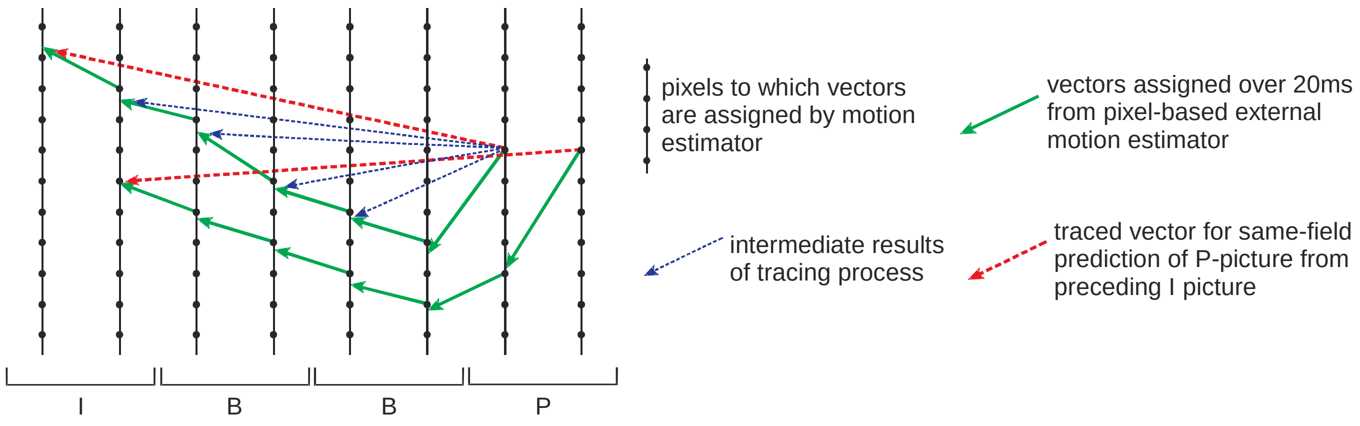
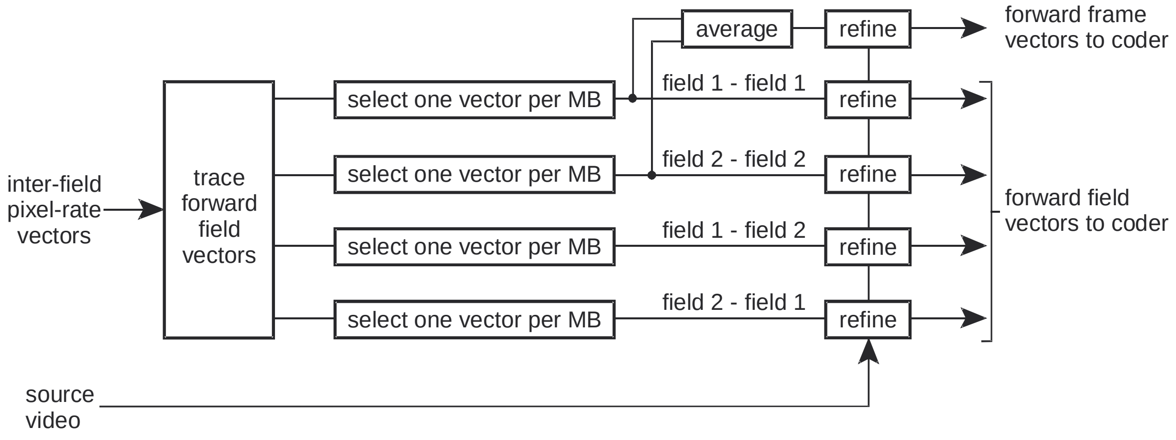
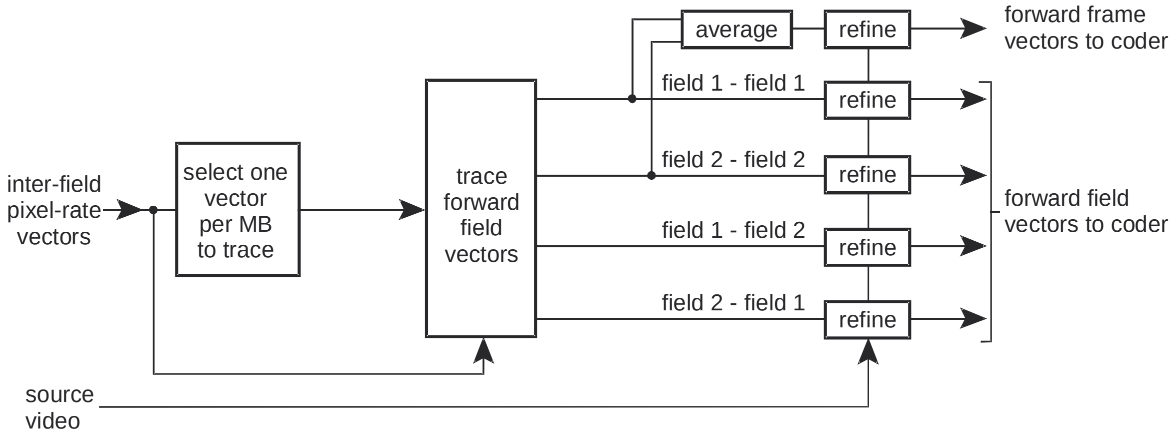

pixels to which vectors are assigned by motion estimator
vectors assigned over 20ms from pixel-based external motion estimator
intermediate results of tracing process
traced vector for same-field prediction of P-picture from preceding I picture
I
B
B
P

Fig. 1 - Tracing field-based vectors to produce GOP-based vectors.



(a) 'Full' tracing.



average
refine
forward frame vectors to coder
inter-field pixel-rate vectors
select one vector per MB to trace
trace forward field vectors
field 1 - field 1
field 2 - field 2
field 1 - field 2
field 2 - field 1
refine
refine
refine
refine
forward field vectors to coder
source video

(b) Tracing one vector per macroblock.

Fig. 2 - The processes involved in the two tracing options considered.

the same type. Note that this algorithm works for 25 Hz film material as well as video, as long as the incoming vector fields are zero between fields of the same video frame.

A particularly useful feature of the vector tracing method is that the maximum displacement that can be measured inherently increases in proportion to the time period between the predicted and reference frames. This is simply because the number of vectors added together is equal to the number of field periods spanned by the vector. Conversely, an approach which estimates motion directly between predicted and reference fields needs a search area which expands with the time period spanned by the vector; this results in the need for very large search areas, particularly between P pictures. Many MPEG-2 hardware encoders therefore have maximum motion ranges that are lower than is desirable, resulting in coding impairments with rapidly-moving picture material.

2.2 Simulation results

2.2.1 Full-resolution tracing for all pixels followed by block formatting

The first experiments examined the performance of the most computationally-intense form of vector tracing, in which vectors are traced for each pixel of the incoming vector field. Vectors appropriate for each macroblock were then derived by computing the most frequently-occurring vector within each macroblock in the traced vector fields. The test sequences used are described in the Appendix. A software simulation of MPEG-2 encoding at main profile, main level was used, with a 12-frame GOP and two B pictures between P pictures using 'frame picture' mode. Further details of the simulation software, which was developed in the *COUGAR* project, can be found in Ref. 6.

Experiments were carried out to compare the coding performance of the traced vectors with those from a full-search block-match. The search range of the block-match was chosen individually for each sequence, to encompass the range of actual motion present; these ranges are listed in the Appendix. The range of the traced vectors was theoretically limited only by the range of the incoming inter-field vectors, which was ±128 pixels and ±128 picture lines per frame. In practice, however, the traced vectors were limited to ±63 pixels and ±28 picture lines per frame, to simulate limited word-lengths in a hardware implementation. The actual maximum-allowed motion vector size was equal to these figures multiplied by the number of frame periods between the predicted and reference frames. The coder bit-rate control was set to 3 Mbit/s.

The results of these first experiments were reported in Ref. 4. It was found that using the traced vectors directly for coding produced comparatively poor results. For example, the peak signal-to-noise ratio (PSNR) was almost 2 dB worse than from a full-search block-match for the *basketball* test sequence for a given total bit-rate. However, the bit-rate required to transmit the vectors themselves was reduced by just over a factor of two, due to their high degree of uniformity. A coding performance that was very similar to that from a full-search block-match was obtained by using the vectors derived from tracing to guide a small-area block-match having a range of ±1 or ±2 pixels (0.24 dB and 0.09 dB worse respectively). This showed that the inaccuracies in the traced vectors were small. These inaccuracies were of the approximate magnitude that would be expected, given an incoming vector field with an accuracy in the region of a quarter of a pixel per field period, being integrated for up to six field periods (the period between P pictures).

The results reported in Ref. 4 also showed that the traced vectors performed better than those from a full-search block-match for a sequence containing a fade to black, due to the inherent immunity of the phase correlation method (and hence the vectors being traced) to changes in DC level. It was also shown that the traced vectors could be used to perform good quality field-rate up-conversion for display.

2.2.2 Comparison between 'full-resolution' tracing and tracing one vector per macroblock

The results obtained from initial simulations of the 'full-resolution' tracing approach[4], suggested that tracing inter-field vectors, and refining these with a small-area block-match, provided vectors suitable for MPEG-2 coding which retained the advantages of smooth 'true' vector fields. Therefore, the more computationally efficient form of the process was studied, in which a vector representative of each macroblock is selected *before* tracing, to see whether a similar level of performance could be obtained with much less computation. Importantly, if only one vector per macroblock is to be traced, it is feasible to implement real-time vector tracing using hardware based on microprocessors intended for digital signal processing (DSP). Conversely, special-purpose hardware would be needed to implement the full-resolution tracing approach.

In order to select a vector representative of each macroblock, a small modification was made to the phase correlation motion estimator to make the vector assigned to the middle pixel of each block a suitable choice. This was achieved simply by ensuring that the aperture, over which displaced field differences were calculated when choosing between candidate vectors, exactly matched the area of a macroblock. This parameter is programmable in the hardware version of the motion estimator, and in any case is usually given a value close to that of the area of a macroblock. Thus, the need for computing the most frequently-occurring vector in the block was removed, giving further hardware savings.

The representative vectors were traced in exactly the same way as previously, with the original full-resolution vector fields being available for the tracing process. Sub-pixel accuracy was maintained during the tracing computations.

The results obtained when tracing one vector per macroblock were compared to those obtained when a vector was traced for every pixel. The comparison was made for three levels of refinement (0, 1 and 2 pixels). Three test sequences were used (*mobcal*, *basketball* and *horses*), using the coder in an open-loop mode with fixed quantiser step size, chosen to give a bit-rate in the range 3–4 Mbit/s (a relatively low rate being chosen to place a high importance on the quality of the motion vectors). In this mode, the main indication of coder performance is the bit-rate used. As Table 1 shows, the signal-to-noise ratios of the decoded sequences differed by no more than 0.26 dB for the unrefined vectors and 0.014 dB for vectors with any degree of refinement. The bit-rates were generally slightly higher for the subsampled traced vectors, but the differences became small as the degree of refinement increased. Full-resolution tracing, followed by selection of one vector per macroblock, was thus slightly better on average, the differences were small, particularly when ±2 pixel refinement was used. Consequently, the significant hardware savings that are produced by using subsampled tracing appeared to be justified. All further experiments were therefore conducted using the approach in which one vector per macroblock was traced.

Table 1: Total bit-rate and PSNR of sequences coded using subsampled traced vectors compared to vectors from full tracing followed by selection of one vector per macroblock (positive numbers indicate higher bit-rates and PSNRs for subsampled tracing).

	No refinement		**±1 pel refinement**		**±1 pel refinement**	
	%	**dB**	**%**	**dB**	**%**	**dB**
Mobcal	+13	–0.001	+4.1	–0.012	–0.02	+0.003
Basketball	–1.1	+0.26	+0.72	–0.014	+0.06	–0.005
Horses	+4.0	+0.17	+0.9	–0.002	+0.74	–0.005
Average	+5.3	+0.14	+1.9	–0.009	+0.26	–0.002

It is worth noting that further hardware savings are possible by spatially subsampling the incoming vector fields; this reduces the memory requirements of the DSP used to implement vector tracing. During the tracing process, the rounding of vectors to the nearest available pixel may then introduce a shift of several pixels, but this is only likely to affect the results near object boundaries. In any case, predictions in such areas tend to be poor, since the motion field is generally not uniform within a macroblock.

Some experiments were conducted to assess the likely impact of vector subsampling. To make any effect clearly visible, a large degree of subsampling was tried, storing only one vector per macroblock (subsampling by a factor of 16 horizontally and vertically, compared to the original luminance resolution). Tests were conducted on the *basketball* and *horses* sequences with the coder again running open-loop. With no vector refinement, sub-sampling the vectors increased the bit-rate of the coded picture by 1.1% for *horses* and 0.41% for *basketball* compared to no subsampling. The bit-rate penalty decreased as the degree of refinement increased, reducing, for example, to 0.07% for ±2 pixel refinement for *horses*; subjectively, the use of vector subsampling made no significant difference. This degree of subsampling is much larger than would be contemplated in practice, yet produced a negligible effect on the coding efficiency. So it seems reasonable to assume that subsampling the incoming vectors by, for example, a factor of two horizontally and vertically would not adversely affect the performance of the coder, yet would allow a significant saving in DSP memory. Note, however, that vector subsampling was not used in further simulation work.

2.2.3 Detailed assessment of vector tracing and refinement

The performance of the algorithm which traced one vector per macroblock was studied in more detail on a wider range of test sequences, and compared with that of full-search block-matching.

The coder software was again run open-loop, with fixed quantiser step sizes chosen to give a bit-rate around 3–4 Mbit/s. In this mode, the PSNR of the coding process remained approximately constant (within 0.1 dB), and the bit-rate used thus gave an approximate indication of the coding performance using the vectors. An alternative approach would have been to hold the bit-rate constant using a rate control algorithm, and examine the PSNR of the coding process; however this approach does not work well on short sequences, since the rate control algorithm does not have long enough to settle down. ‘Intra’ coding in P and B pictures was disabled so that prediction errors fully reflected the performance of the vectors, making prediction error signals easier to interpret and more informative. Experiments were performed with 0, ±1 and ±2 pixels refinement as in the earlier work, as well as half-pixel refinement on its own. The details of these refinement processes were:

- **unrefined**: the vectors directly from the tracing process, rounded to the nearest half-integer value.
- **half-pixel refinement**: the traced vector was rounded down to the nearest integer value, and the mean modulus displaced-field differences were calculated for this vector, and the eight vectors offset from this by ±½ a pixel and ±½ a **frame** line (for frame prediction), or ±½ a **field** line (for field prediction). The vector giving the lowest difference was chosen.
- **one pixel plus half-pixel refinement**: the traced vector was rounded down to the nearest integer value, and vectors offset from this by ±1 pixel and ±1 **frame** line (for frame prediction) or ±1 **field** line (for field prediction) were tested; a half-pixel refinement as above was then carried out on the vector giving the lowest displaced field difference.
- **two pixel plus half-pixel refinement**: as above, but the search range for the integer search was extended to ±2 pixel and ±2 picture lines for frame prediction (for field prediction, the vertical search range was ±1 field line).

Fig. 3 shows the bit-rates measured for the six sequences, as a percentage of the rate using full search block-match vectors (the sequences have been placed in order of increasing performance of the traced vectors relative to block-matching). It can be seen that these results confirm those presented earlier[4]: the unrefined traced vectors performed poorly compared to the block-matching vectors for the first four sequences. For the last two sequences, where the presence of a fade contributes to much (*crossfade*) or virtually all (*threads*) of the temporal changes, the unrefined traced vectors performed similarly to, or much better than, block-matching, respectively.

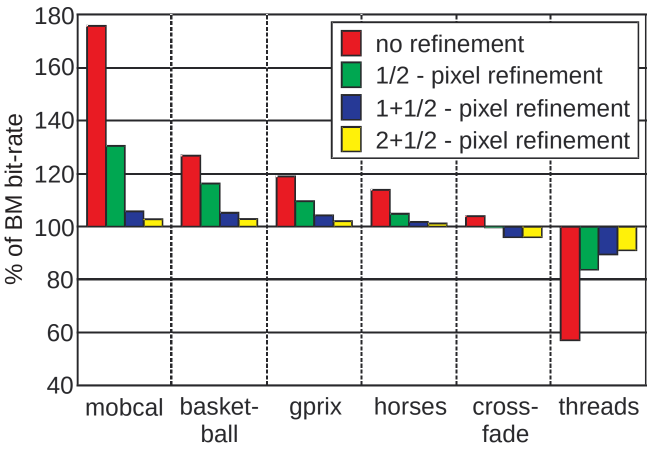


Fig. 3 - Total bit-rates obtained using vectors derived by tracing vectors from phase correlation, expressed as a percentage of the rate using block-matching vectors.

There were several reasons for the poor performance on the first four sequences of the traced vectors with little or no refinement, compared to full-search block-matching:

- Inaccuracies in the traced vectors in uniformly-moving areas (e.g. the background in *mobcal* and *basketball*), due to accumulated inaccuracies in the vectors. This was by far the most significant cause of the poor performance for the traced vectors with no refinement. These inaccuracies were almost entirely removed by ±1 pixel refinement, and the ±2 pixel refinement reduced the prediction errors to levels essentially identical to those from the full-search block-match.

- Inability to track rotating/deforming or small objects (eg. the ball in *mobcal*, some of the players in *basketball*). This is a characteristic of the phase correlation algorithm as implemented in Ref. 2, and is mainly due to the relatively large block size employed in the correlation process (the algorithm was optimised for the measurement of very fast motion). The effect of this was to increase the noise level in corresponding areas in the decoded pictures. Subjectively, this was not particularly annoying, since the areas tended to be small and difficult for the eye to follow.

- Inability to select vectors pointing to similar-looking picture material for revealed background in P pictures (e.g. the revealed material on the right hand side of the image in *horses* and *gprix* as the camera pans, and behind the cars in *gprix*). This is a fundamental limitation of any motion estimator claiming to make 'true vectors', since there is no valid 'true' vector pointing backwards to previous frames for such areas. However, a motion estimator that simply generates 'best match' vectors can attempt to produce a reasonable prediction from some other part of the image. The effect of this shortcoming was to introduce some coding noise in areas of revealed background. In practice, the noise was hardly visible, due to the effect of temporal masking.

- Inability to find vectors for material obscured and revealed again (such as the area over which the ball passes in *basketball*). This is a fundamental limitation of the tracing method: once an area has been obscured, its motion cannot be followed, even if it re-appears in other frames. It is not a limitation of the general concept of using 'true' vectors, though, since there is a valid true vector for such areas. However, since the coding noise generated in such areas is very difficult to see, it was not considered to be a significant problem.

It should be noted that intra coding was disabled in P and B pictures as stated earlier; had this been enabled, then many of the poorly-predicted areas may have been coded as intra blocks. It is common practice for coders to choose intra coding when the energy in the prediction error signal is higher than that in the original signal, since this gives the highest coding efficiency. This would be particularly likely to happen when the poorly-predicted areas contained little detail, as was the case with revealed background during fast camera pans. The subjective appearance of these areas would therefore be improved.

The addition of refinement produced a significant improvement to the first four sequences: the average bit-rate was reduced from 34% above blockmatching to 15% above for half-pixel-only refinement, and to 4.2% above for ±1pixel. A further improvement to 2.2% above blockmatching was achieved by increasing the refinement search area to ±2 pixels. This confirmed that the most significant problem with the traced vectors was simply the small degree of accumulated inaccuracy.

For *crossfade* and *threads*, the picture quality was better than from block-matching, as vectors were not distorted to find brighter or darker areas to compensate for the brightness change. An example of the vectors

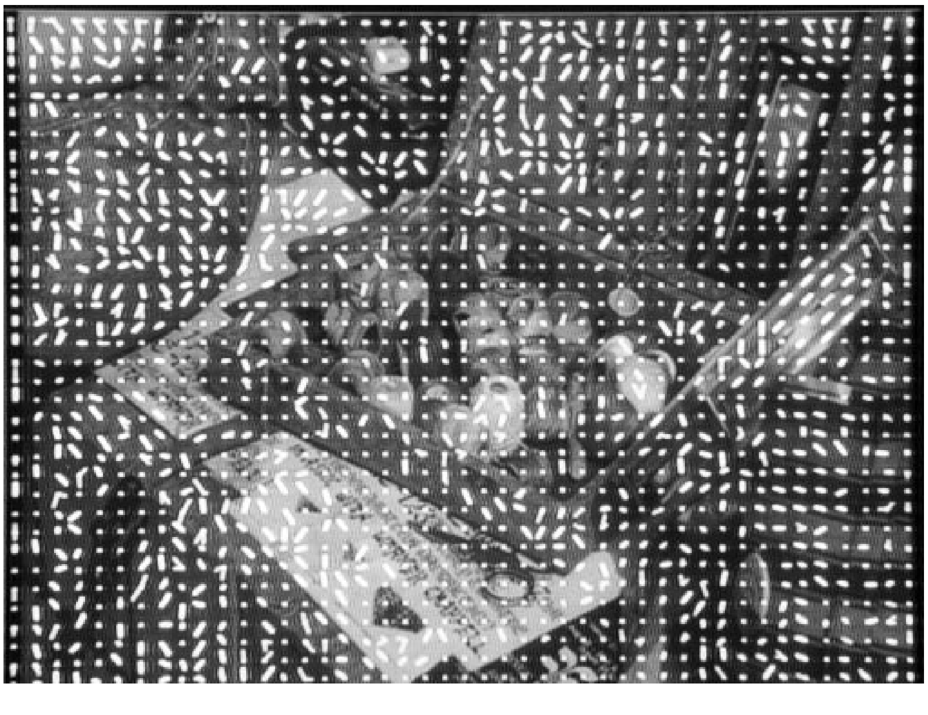


Fig. 4 - Motion vectors generated by full-search block-matching for a stationary image showing a fade to black.

generated by block-matching for one frame of *threads* is shown in Fig. 4. The vectors are superimposed on the original image, which has faded to near-black at this point in the sequence. The vector for each macro-block is shown as a line originating from the centre of the block, having a direction and length indicating the estimated motion per field period. It can be seen that many vectors have large values despite the image being stationary, due to the block-matcher attempting to find an area of similar brightness in the adjacent frame. Conversely, the vectors generated by phase correlation and tracing remained essentially zero throughout the image during the sequence. As might have been expected, refinement had a detrimental effect on the *threads* sequence, since it allowed vectors to begin to follow the mean DC level (rather than the motion), leading to poorer predictions for all spatial frequencies except DC, and an increased vector bit-rate.

2.2.4 Summary of investigations

These experiments showed that the vector tracing algorithm, followed by ±1 or ±2 pixel refinement, gave a coding performance close to that from an exhaustive search block-match for the four test sequences not containing fades. The bit-rate was an average of 4.2% higher than block-matching with ±1 pixel refinement, and 2.2% higher for ±2 pixel for these sequences. Given that an exhaustive search block-match is impracticable to implement in hardware (particularly for the large search range needed for *gprix* of ±180 pixels between P pictures), and phase correlation, followed by vector tracing and refinement, are clearly practicable, this in itself can be considered a useful result.

Furthermore, for sequences containing fades or cross-fades, the vectors derived from tracing performed significantly better than those from block-matching, due to the inherent immunity of the phase correlation method (and hence the vectors being traced) to changes in DC level. However, the vector refinement process tended to have a detrimental effect on such sequences, since it allowed vectors to be distorted away from the true motion by attempting to match each block to one with the same overall brightness. Thus, some method of improving the refinement process so as to maintain the full benefit of 'true' vectors during fades was investigated, and is described below.

3. MODIFICATIONS TO VECTOR REFINEMENT TO COPE WITH FADES

3.1 Possible solutions to the fade problem

Perhaps the most obvious approach to the detrimental effect of vector refinement during fades would be to detect the presence of a fade, and disable the refinement process accordingly. However, the reliable detection of fades is difficult, and furthermore this approach would not cope with brightness changes over parts of the image, for example due to changes in lighting. Therefore, this approach was not considered further.

An alternative approach is to make the small-search-area block-matching process inherently 'blind' to brightness changes. One way of achieving this is to subject the video signal to a spatial high-pass filter prior to block-matching, in order to remove low spatial frequencies. Ideally, the filter should have a sharp null at DC so as to remove as little detail as possible whilst removing all the DC component. However, hardware constraints favour small filters which will inevitably have relatively broad nulls.

Another option is to measure and subtract the mean DC levels from the block in the reference and predicted images during the block-match process itself. Normally, a block-matching process seeks to minimise the cost function, as:

$$\sum \left| y_{mb} - y_{ref} \right|$$

where y_{mb} is the luminance level at a pixel in the macroblock to predict, and y_{ref} is the luminance level in the reference image, at a position offset by the vector being assessed, and the summation is over all pixels in the macroblock. When the mean DC levels are removed, the cost function becomes:

$$\sum \left| \left(y_{mb} - DC_{mb} \right) - \left(y_{ref} - DC_{ref} \right) \right|$$
$$= \sum \left| y_{mb} - y_{ref} - d \right|$$

where $d = DC_{mb} - DC_{ref}$

$$DC_{mb} = \sum y_{mb}$$
$$DC_{ref} = \sum y_{ref}$$

The DC level of the block in the image being predicted need only be measured once for each block; however the DC level of the block in the reference image should ideally be measured for every search position. Significant computation could be saved by measuring an average DC level over the whole of the search range in the reference image, although this might be expected to reduce the performance. This latter approach will be referred to as 'fast DC-removal'.

3.2 Simulations of refinement using high-pass filtering and DC-removal

The performance of various options was assessed by simulation, using the same test sequences and coding method used to compare degrees of refinement. Fig. 5 *(overleaf)* shows the results in terms of percentage bit-rate change compared to refinement on the unprocessed source signal, for refinement by ±2 pixels followed by half-pixel refinement (both refinement

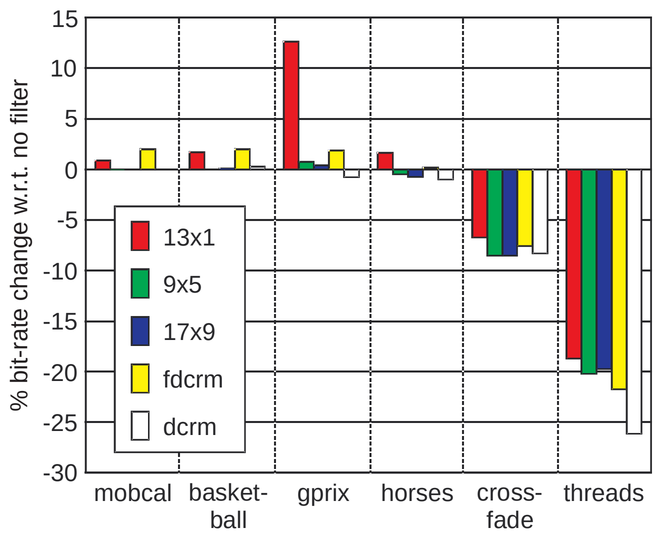


Fig. 5 - The performance of three different high-pass filters and two DC-removal methods for vector refinement.

stages using the same filtering or dc-removal method). The first three results for each sequence are for the use of a high-pass filter on the source signal prior to the block-match refinement process. The three filters had different apertures, the larger apertures giving a sharper null at DC but requiring more hardware. The apertures (in units of pixels × field lines) were 13 × 1 (i.e. horizontal only), 9 × 5 and 17 × 9. The penultimate result (labelled 'fdcrm', meaning 'fast DC-removal') refers to the use of a DC-removal technique within the block-match process, in which the average DC level for the whole search area in the reference image is used. The final result refers to DC-removal in which a DC level is computed separately for each search position.

The most important conclusion that can be drawn from the results is that all the high-pass filtering and DC-removal techniques show a significant benefit for the two sequences that contain significant brightness changes (*crossfade* and *threads*, the last two shown in the figure). In addition to needing fewer bits to code, these two sequences looked subjectively much better, with little or no loss of resolution or introduction of noise during the brightness changes.

For the sequences not containing fades (the first four in the figure), the one-dimensional high-pass filter performed poorly compared to the use of no filter, particularly on *gprix*. This was thought to be because this sequence contained little spatial detail in many areas (due to blur from the rapid motion speed). Thus, the removal of a large part of the spectrum (low horizontal frequencies for all vertical frequencies) significantly reduced the information available to the directed block-matcher. However, refinement on a signal that was filtered, using **either** of the two-dimensional high-pass filters, gave results that were, on average, very similar to refinement on **un**filtered video, for the non-fading sequences; the results were slightly better on *horses*, which contains slight brightness changes, and marginally worse on *gprix*, which is soft.

The fast DC-removal technique performed slightly worse than conventional block-matching refinement for the non-fading sequences, giving an average bit-rate 1.5% higher. But the full DC-removal method showed no such effect, and gave an average bit-rate for the non-fading sequences that was actually 0.4% *better* than from conventional block-match refinement. It also gave the best performance on the fade-to-black (*threads*).

Therefore, out of the methods studied, the full DC-removal method appeared to give the best performance, and achieved the aim of preventing a block-match-based vector refinement process from destroying some of the benefits of the 'true' motion vectors derived by phase correlation and tracing. Furthermore, implementing DC-removal within the block-match refinement process was more attractive for hardware implementation than providing a separate high-pass filter and associated additional routeing of high-pass-filtered video.

3.3 High-pass filtering applied to full block-matching

Having shown that the techniques of high-pass filtering or DC-removal improve the performance of block-matching when used to refine traced vectors, the question naturally arises as to whether such techniques would improve the performance of conventional full-search block-matching. Some brief experiments were carried out to answer this question.

The exhaustive-search block-match software, that was used to derive the vectors for the comparisons reported above, was re-run on signals that had been subjected to a two-dimensional high-pass filter. The resulting coder bit-rates, expressed as a percentage of the bit-rate when conventional full-search block matching was used, are shown in Fig. 6. It can be seen that the use of a high-pass filter increased the bit-rates for the first four non-fading sequences slightly (by an average of 1.0%) whilst significantly decreasing the bit-rate for the last two sequences which contain fades (by an average of around 20%). Indeed, the performance of block-matching with the high-pass filter on the fading sequences is essentially the same as that from the traced and refined vectors.

Nevertheless, whilst the use of a high-pass filter prior to conventional block-matching appeared to overcome one deficiency of block-matching (sensitivity to brightness change), the extent to which the vectors represented true motion was found, not surprisingly, to be left largely unchanged. A typical field from the *horses* sequence is shown in Fig. 7 *(page 10)*, with estimated vectors indicated by small lines as in Fig. 4. The lack of correlation between estimated and actual motion is clearly visible in both the vector fields derived by block-matching (Fig. 7(a) and (b)), regardless of the presence of a high-pass filter. The uniformity of the vector field and its correlation with moving objects is, however, clearly visible in the vectors derived by tracing and ±2 pixel refinement, Fig. 7(c). The vectors from either type of block-matching method in this example would clearly be of little use for applications such as display up-conversion.

4. FURTHER CONSIDERATION OF HALF-PIXEL REFINEMENT

4.1 Why half-pixel refinement should be considered separately

In the work reported in Section 3.2, the DC-removal method was used for both the integer and half-pixel refinement stages. It is possible that the main benefits of DC-removal occurred in the integer refinement stage, and that using it for half-pixel refinement might add unnecessary complexity to the hardware. So further experiments were carried out to compare the presence and absence of DC-removal at the half-pixel refinement stage, given vectors that had undergone integer refinement using DC-removal. It should be noted that only the 'fast' type of DC-removal (explained earlier) was used at the half-pixel refinement stage. This was deemed to be a reasonable simplification; since the search area is very small (±½ pixel) compared to the size of a macroblock (16 × 16 pixels), the mean DC level would not be expected to change significantly for different search positions.

Furthermore, in all the simulation work described above, the **source** picture was used for all motion estimation processes, including the half-pixel refinement stage, since this prevented coding artefacts from influencing the motion vectors and should result in vectors that correspond more closely to the true motion. However, it is often considered advantageous to use the **locally-decoded** picture for motion estimation, and in particular, for half-pixel refinement. Experiments were therefore conducted to determine whether use of the locally-decoded picture for half-pixel refinement (whilst retaining the source signal for the integer refinement stage) might offer a better option.

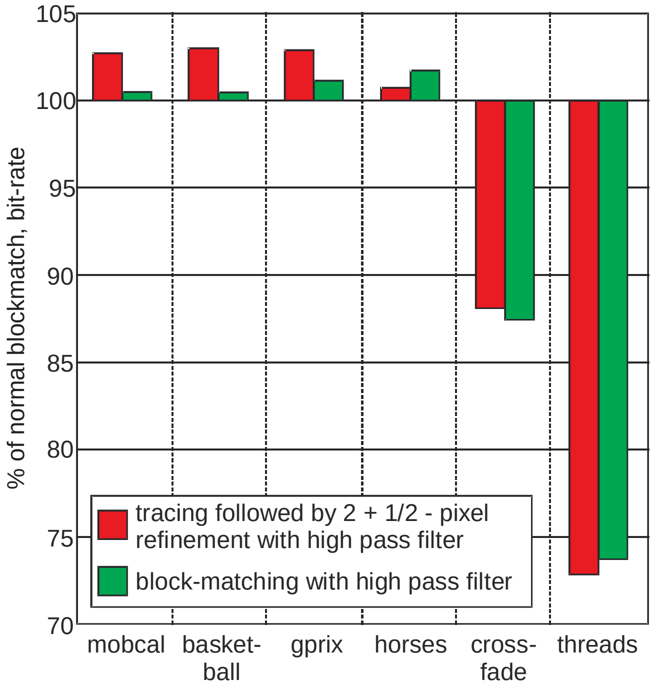


Fig. 6 - Bit-rates for coding test sequences using traced vectors that were refined using a high-pass filtered source, and full-search blockmatch vectors using a high-pass filtered source; both compared to full-search blockmatching using the unfiltered source.

As with the previous work, the same six test sequences were used, with the coder running open-loop with the quantiser step size fixed. Vectors derived by phase correlation were traced, and refined to the nearest integer position using a directed block-match incorporating DC-removal, with a ±2 pixel search range. Four half-pixel refinement strategies were then tested: with or without DC-removal, and using either the source or locally-decoded picture. The bit-rates and PSNR values for each sequence under the four conditions were calculated, and are shown in Figs. 8 and 9 *(page 11)* respectively, compared to the figures for no half-pixel refinement.

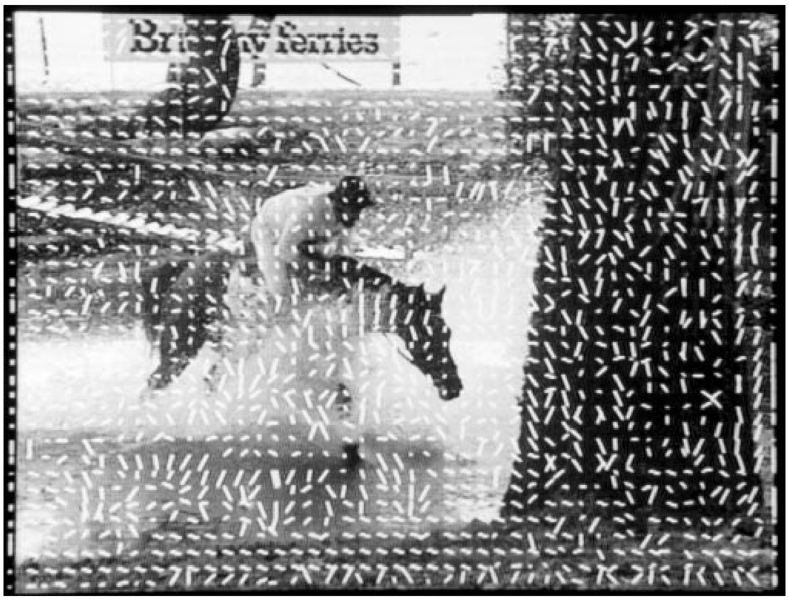
4.2 DC-removal for half-pixel refinement

It can be seen that DC-removal had little effect on either the bit-rate or the PSNR of first four (non-fading) sequences when using either the source or locally-decoded picture. For the last two sequences (that contain fades), DC-removal decreased the bit-rate whilst increasing the PSNR. The improvement on *threads* was most significant, particularly when using the source picture for refinement (a saving of 30% in bit-rate, accompanied by a significant improvement in subjective appearance). Indeed, although refinement without DC-removal gave an increase in PSNR compared to no refinement, the subjective effect was a significant **worsening** in picture quality. These results show that DC-removal plays an important part in half-pixel refinement, as well as in the preceding integer-pixel refinement stage, and should be included in the hardware being developed.

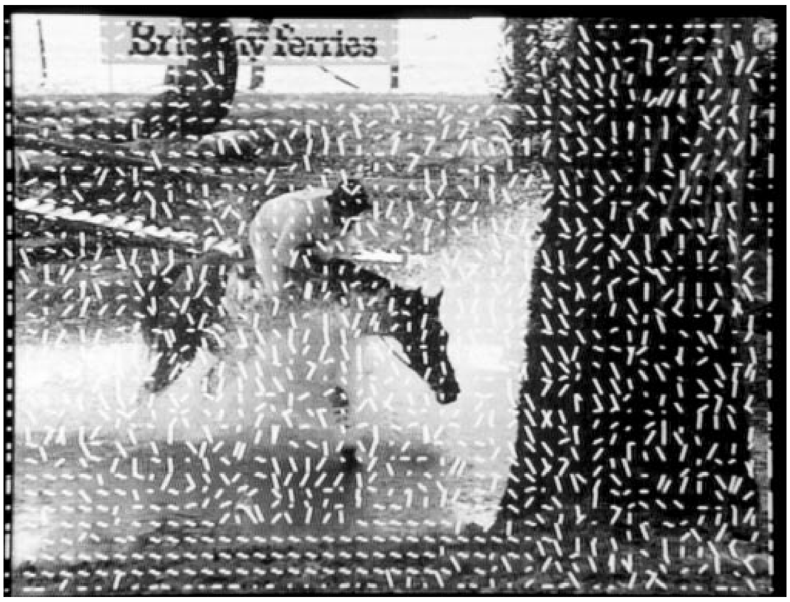
4.3 Half-pixel refinement using locally-decoded or source pictures

It can be seen from Figs. 8 and 9 that refining, using the locally-decoded signal rather than the source, tended to increase the bit-rate – by an average of about 3% on the first five sequences. This was accompanied by an increase in the PSNR (an average of about 0.15 dB on the first five sequences). In the case of the sixth sequence, *threads*, the bit-rate increase was much larger.

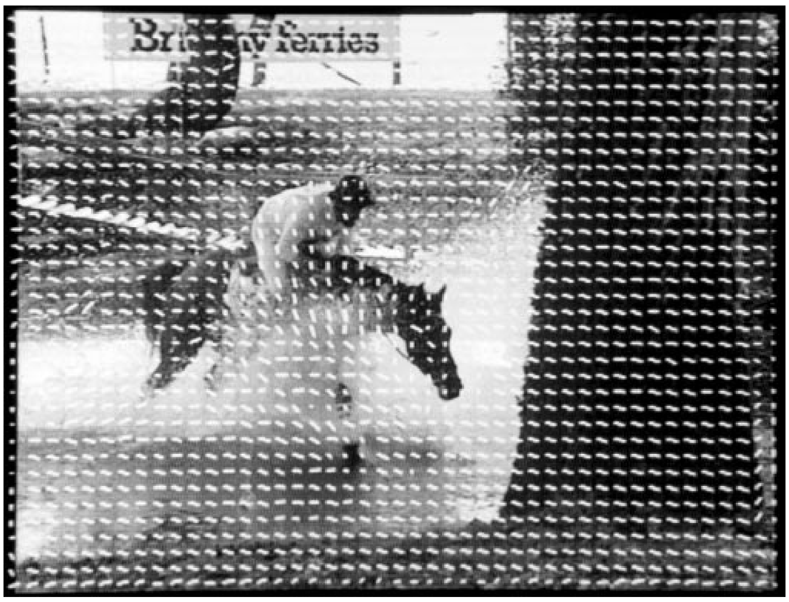
Further investigation showed that the increase in bit-rate was due entirely to more bits being needed to code the vectors. This was because the vectors tended



(a) Conventional block-matching



(b) Block matching on a high-pass filtered image.



(c) Tracing and refinment of vectors derived by phase correlation.

Fig. 7 - Motion vectors for one field of the 'horses' test sequence.

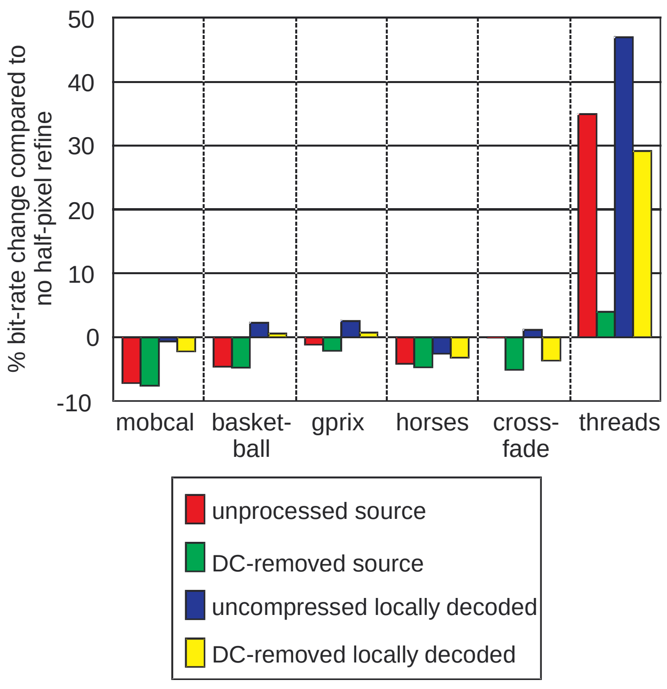


Fig. 8 - Effect on bit-rate of DC-removal and source vs. locally decoded picture for half-pixel refinement.

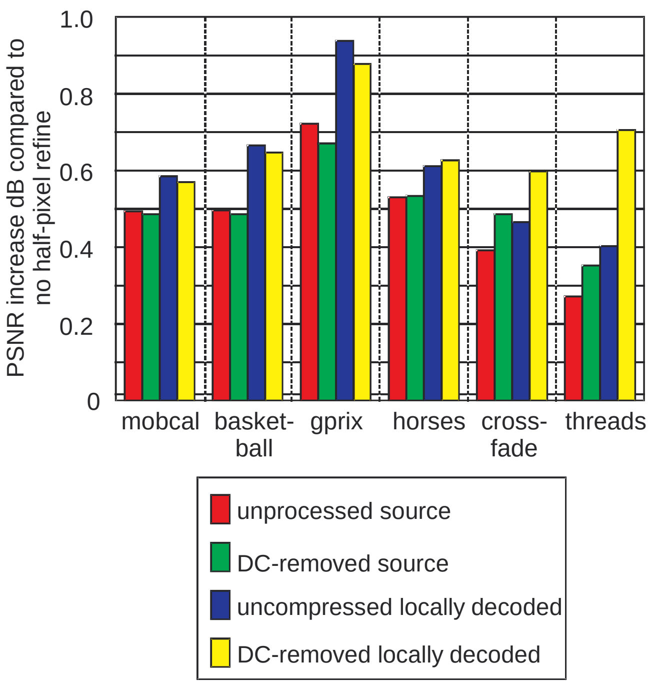


Fig. 9 - Effect on PSNR of DC-removal and source vs. locally-decoded picture for half-pixel refinement.

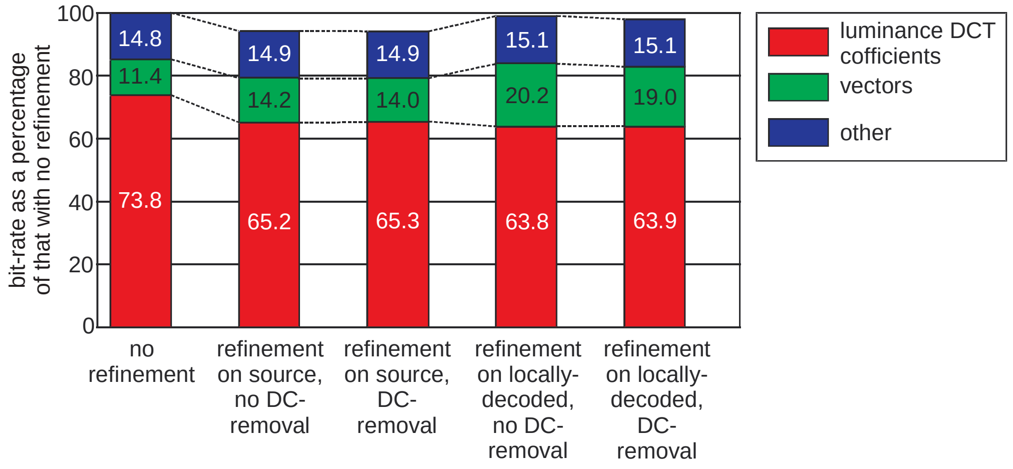


Fig. 10 - Breakdown of bit-rate for basketball, for no half-pixel refinement, and refinement with and without DC removal on source and locally-decoded pictures.

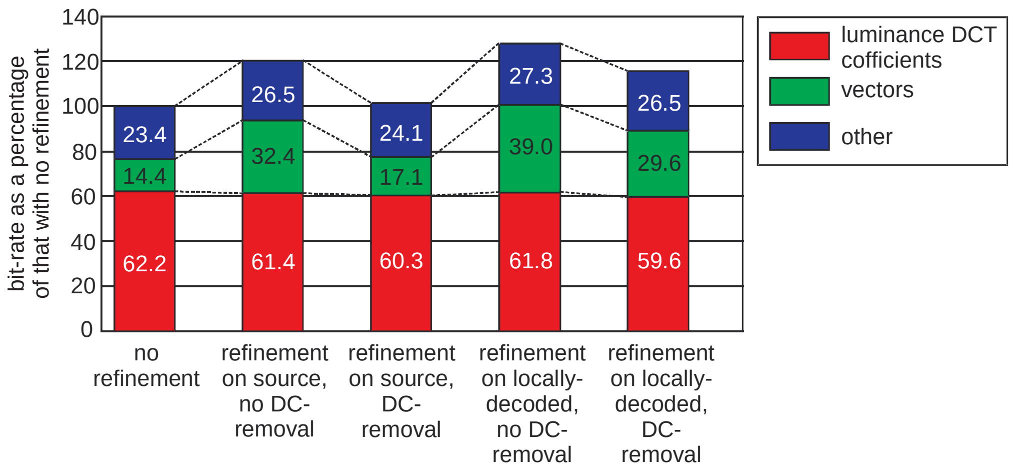


Fig. 11 - Breakdown of bit-rate for threads, for no half-pixel refinement, and refinement with and without DC removal on source and locally-decoded pictures.

to vary by small amounts from block-to-block as they attempted to follow the coding noise. This variation reduced the efficiency of the differential encoding algorithm that is used in MPEG-2 to code the vectors in the bitstream. The number of bits required to code the luminance was reduced (since a slightly better prediction was generated), but not by enough to compensate for the increase in vector bit-rate. This can be seen in Figs. 10 and 11 *(previous page)*, which show the breakdown in bit-rate between motion vectors, luminance DCT coefficients, and other data for the sequences *basketball* and *threads*. The figures in the bars indicate the bit-rate used, and are expressed as a percentage of the total bit-rate for the sequence **without refinement**.

Subjectively, the increase in PSNR from using the locally-decoded picture for half-pixel refinement was not discernable. Indeed, the only significant effect of locally-decoded refinement was seen on *threads*, where the noisy vector field created small random movements in some areas of the image, causing a small subjective **degradation**, despite the measured improvement in PSNR.

It was not straightforward to make a judgement between source and locally-decoded refinement using the objective measures of bit-rate and PSNR, since both of these measures showed small increases (conversely, in previous experiments, using a fixed quantiser step size, the PSNR remained virtually unchanged, so the bit-rate alone provided a useful performance measure). One way to resolve this would have been to conduct experiments which included a rate-control algorithm to force a constant bit-rate; however, this would have introduced further problems as explained earlier. An alternative method was to measure the relationship between bit-rate and PSNR when the quantiser step size was changed and all other parameters were held constant. This relationship was studied using the *mobcal* test sequence for bit-rates in the range 3–6 Mbit/s, and found to be approximately 0.05 dB per percent increase in the bit-rate. This result suggested that the average PSNR increase of 0.15 dB that was brought about by the use of locally-decoded pictures could alternatively have been achieved by simply changing the quantiser step size so as to increase the bit-rate by 3% – the same as the average increase measured when locally-decoded pictures were used.

Thus, the overall effect of refining by using the locally-decoded picture appeared to be neutral when using the PSNR and bit-rate as performance measures – at least, with the relatively uniform vector fields produced by the motion estimation method described here. This conclusion may, of course, have been different if the vectors being refined had been less uniform in the first place; the coding algorithm used to code the vectors might already have been working with a lower efficiency, so that the net increase in the vector bit-rate from locally-decoded refinement might have been smaller.

5. CONCLUSION

This Report has considered the conversion of a ‘true’ inter-field pixel-rate motion vector signal into a form suitable for use by an MPEG-2 coder. The performance of a coder using the vectors has been assessed, and compared to that of a coder using vectors derived by an exhaustive-search block-matching algorithm.

The broad conclusions from the work are as follows:

- Vectors from an inter-field motion estimator, based on phase correlation, can be converted successfully for use in an MPEG-2 coder by ‘tracing’ the vectors. However, refinement using a small-search-area directed block-match with a search range of ±1 or ±2 pixels centred on the traced vector is needed in order to obtain the required accuracy. The tracing algorithm requires comparatively little computation, and is amenable to real-time implementation using a DSP.

- The use of the traced vectors with ±2 pixel refinement resulted in bit-rates about 2% above those from a full-search block-match, averaged over four test sequences. For sequences containing fades and cross-fades, a bit-rate saving of up to almost 30% compared to that from full-search block-match vectors was measured, although to achieve this saving, the refinement process had to be modified to ignore DC.

- The half-pixel refinement processing may be carried out either on the source or the locally-decoded video signal, with little overall effect on the performance.

The work described here was carried out as a part of the EU-funded RACE *COUGAR* project. Further work in this project, concerned with the use of vectors recovered from an MPEG-2 bitstream for motion-compensated display conversion, may be found in Ref. 3.

6. REFERENCES

1. THOMAS, G.A., 1987. Television motion measurement for DATV and other applications. BBC Research Department Report No. 1987/11.

2. LAU, H. *and* LYON, D., 1992. Motion compensated processing for enhanced slow-motion and standards conversion. IBC ’92 (Amsterdam), 4–7 July 1992. IEE Conference Publication No. 358, pp. 62-66.

3. THOMAS, G.A., 1996. A comparison of motion-compensated interlace-to-progressive conversion methods. BBC R&D Report No. 1996/9.

4. THOMAS, G.A. *and* DANCER, S.J., 1995. Improved motion estimation for MPEG coding within the *RACE 'COUGAR'* project. IBC '95, IEE Conference Publication No. 413, pp. 238-243.

5. LEE, X., ZHOU, P. *and* LEON-GARCIA, A., 1992. An efficient MPEG motion compensation scheme by motion trajectory tracking method. SPIE **1818**, Visual Communications and Image Processing '92, pp. 594-605.

6. TUDOR P.N. *and* BRIGHTWELL P.J., 1996. COUGAR software model – Architecture for hardware-specific optimisation experiments. Proceedings of the European Workshop on Image Analysis and coding for TV, HDTV and Multimedia Applications (HAMLET workshop), February, Rennes, France, pp. 85-94.

APPENDIX

Test sequences used in the simulation work

The test sequences used in the work reported here are listed below. The first 25 frames of each sequence were processed.

mobcal	The well-known ‘mobile and calendar’ test sequence.
basketball	A well-known MPEG test sequence; the first 25 frames were used, in which the camera starts almost stationary, and then pans to the right.
gprix	A rapid camera pan following Formula 1 racing cars round a bend. Shot with a shuttered CCD camera, recorded in PAL and subsequently decoded. The background motion speed is approximately 30 pixels per field period.
horses	A well-known MPEG test sequence; the 25-frame portion chosen shows a horse jumping into water.
crossfade	A cross-fade between the *basketball* and *horses* sequences, starting a few frames after the start of the sequence and finishing a few frames before the end.
threads	A fade to black on a virtually stationary scene showing some brightly-coloured balls of thread.

The search range of the exhaustive-search block-match motion estimator was chosen to encompass the actual motion speeds believed to be present in each sequence. The search ranges used, in units of pixels and picture lines per frame period, were as follows:

mobcal:	±25 × ±20
basketball:	±25 × ±20
gprix:	±60 × ±30
horses:	±30 × ±32
crossfade:	±30 × ±25
threads:	±20 × ±15

Note that the actual search range for a given frame was equal to these values multiplied by the number of frame periods between the reference and predicted frame. For example, the search range for vectors for a P picture in *gprix* was ±180 pixels × ±90 picture lines.

Published by BBC RESEARCH & DEVELOPMENT DEPARTMENT, Kingswood Warren, Tadworth, Surrey, KT20 6NP